Wavelets

Scattered Data Interpolation using Wavelet Trees

User guide

Jean-Baptiste Keck

M2 Msiam

February 9, 2015

Contents

1 Introduction

This is a partial implementation of Christophe P. Bernard thesis on interpolating Deslauriers and Dubuc wavelets.

The implementation concists in three main parts :

- A maple script to generate low pass filter coefficients h_n and Deslauriers-Dubuc wavelets (1D, 2D).
- Python scripts to generate Deslauriers-Dubuc wavelet families on the interval.
- Final efficient implementation of the method in one dimension written in C++.

2 Folder architecture

Here is an synthetic overview of given files :

- Root folder
  - *readme.pdf* .... This file !
  - *CMakeLists.txt* .... CMake configuration file for C++
  - src/ .... Contains all C++ sources and headers
  - build/ .... Used to compile C++
  - modules/ .... CMake helpers
  - scripts/ .... Contains all maple and python scripts
  - papers/ .... Contains the thesis and paper about implemented method
  - results/ .... Contains some precomputed results
  - slides/ .... Contains slides pdf, tex and images

3 Original papers

The original thesis and paper can be found in the following folder :

- Root folder
  - papers/
    - *paper.pdf* .... Short introduction paper on the proposed method
    - *these.pdf* .... Original PhD Thesis on the method, see pages 151 to 231

4 Slides

Slides of the presentation can be found at the following locations :

- Root folder
  - slides/
    - *slides.pdf* .... Slides in pdf version
    - **.tex* .... Latex sources
    - *master.bib* .... Bibliography
    - img/ .... All images and animations used during presentation

5 Scripts

Here are the concerned files of this section :

- Root folder
  - scripts/ .... Contains all scripts
    - *wavelets.mw* .... Maple script
    - **.py* .... Python scripts

5.1 Maple script

The maple script can generate the filter coefficients h_n with Lagrange polynomials of any given order, it then iteratively compute Deslauriers and Dubuc wavelets with convolutions of the filter with a dirac. Finally it computes their Fourier transform and perform nice plotting. It was only tested with maple 18.

Running the script : ~maple scripts/wavelets.mw

Parameters :

- **Pmin :** Minimal Deslauriers-Dubuc order to generate (default = 1)
- **Pmax :** Minimal Deslauriers-Dubuc order to generate (default = 4)
- **levels :** Number of convolutions used to generate the wavelets (default = 4)

Output overview :

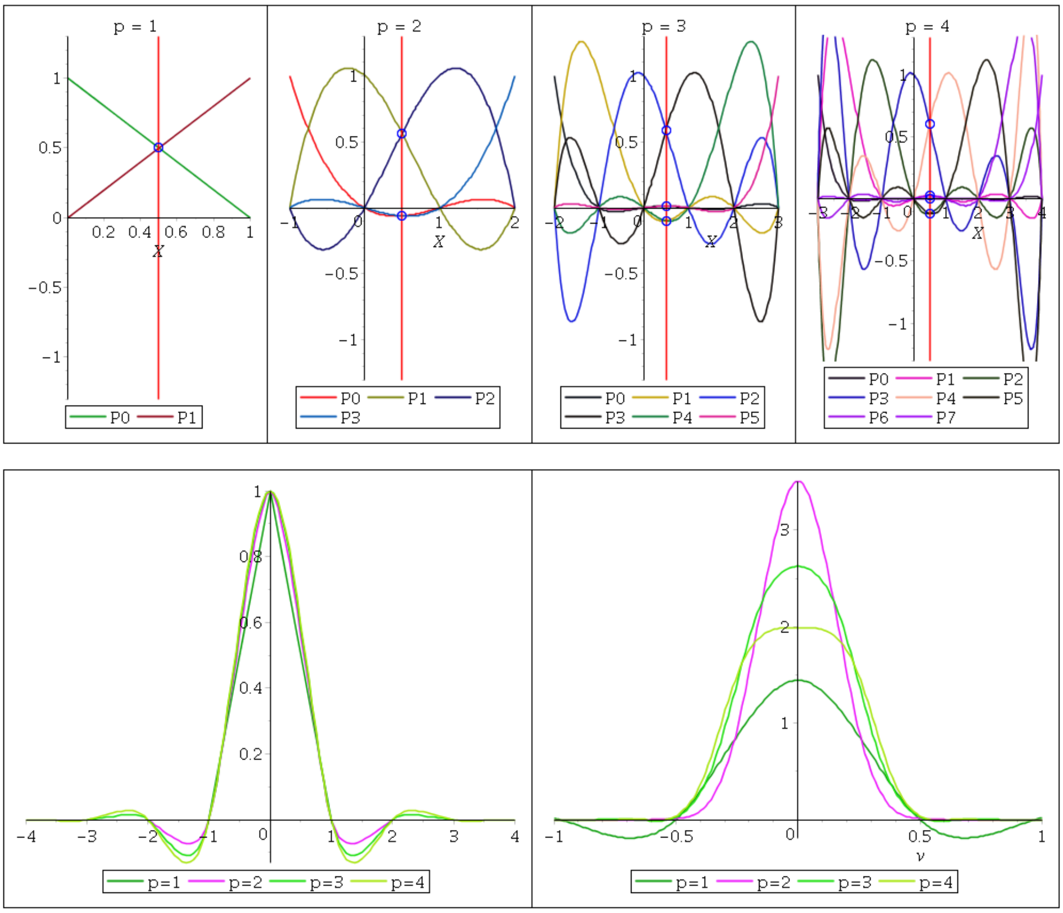

Figure 1: Output of the maple script *wavelets.mw*

5.2 Python scripts

Python scripts can generate Deslauriers-Dubuc wavelet families on the interval as well as Deslauriers and Dubuc wavelet alone. It was coded and tested with python 2.7.6.

There are 3 scripts :

- **scripts/plot.py** : to plot 1D wavelets
- **scripts/plot2D.py** : to plot 2D wavelets
- **scripts/plotFamilies.py** : to plot wavelet families at a given level j on the interval $\Omega = [0, 1]$

Running a script : ~python <script name>

Parameters : See comments in code.

Output overview :

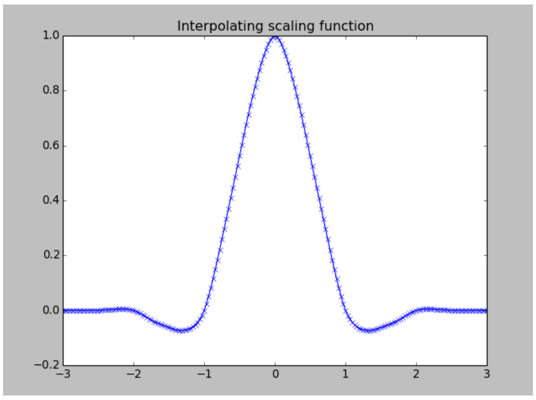

Figure 2: Output of *plot.py*

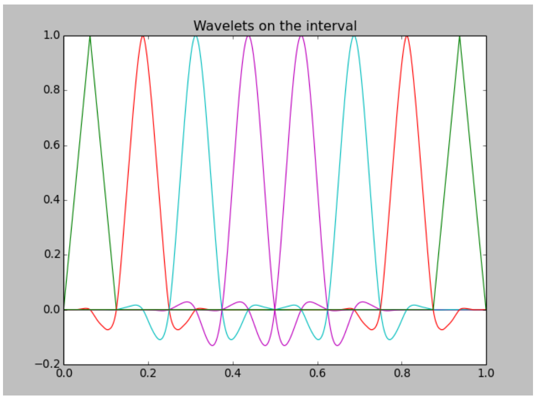

Figure 3: Output of *plotFamilies.py*

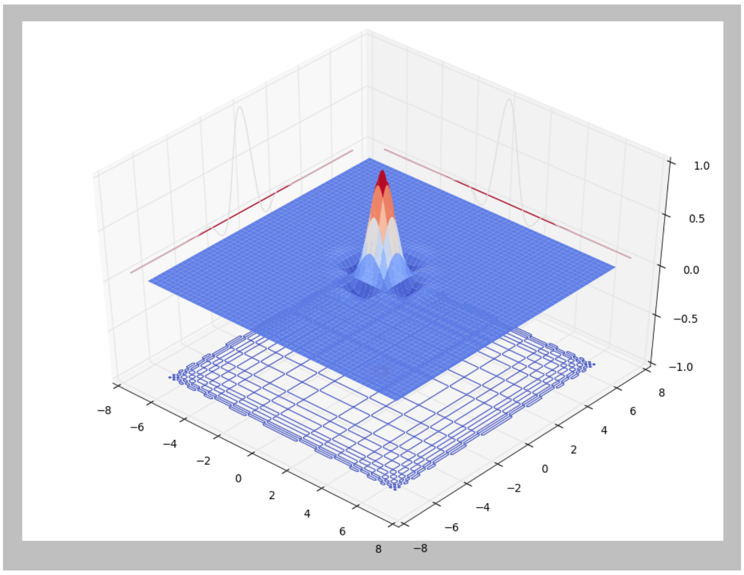

Figure 4: Output of *plot2d.py*

6 Method implementation

The full one dimensional method has been implemented in C++. It consists into three executables :

- **main :** Apply and plot the intermediate steps of the proposed interpolation method.
- **bench :** Benching code to check efficiency and check interpolation error in ℓ_2 and ℓ_∞ norm.
- **animation :** Executable to generate incremental sample interpolation animations (generate gif files).

6.1 Dependencies

To compile you need a Unix like distribution and a C++11 capable compiler : Tested with gcc 4.8.2 on Linux Mint 17 Qiana.

It should work on a Mac but it has not been tested.

The following libraries are required for the compilation of the C++ sources :

- Console Gnuplot (C++ wrapper */src/gnuplot/gnuplot.hpp*)
- Eigen3 (for linear system solving)
- Boost (mainly for the gnuplot wrapper)
  - System
  - Filesystem
  - Iostreams
- ImageMagick (to generate gifs)
  - MagickCore
  - Magick++

6.2 Compiling

The three executables (*main*, *bench* and *animation*) can be generated with the following commands :

In normal mode :

```
~ cd build/
~ cmake ..
~ make -j8
```

In debug mode (add debugging symbols):

```
~ cd build/
~ cmake -DCMAKE_BUILD_TYPE=Debug ..
~ make -j8
```

In release mode (enable optimizations):

```
~ cd build/
~ cmake -DCMAKE_BUILD_TYPE=Release ..
~ make -j8
```

Executables will be generated directly in the **build/** directory.

Execution : ~ ./<executable name>

6.3 Main program

Apply and plot the intermediate steps of the proposed interpolation method. See comments in the code.

Overview :

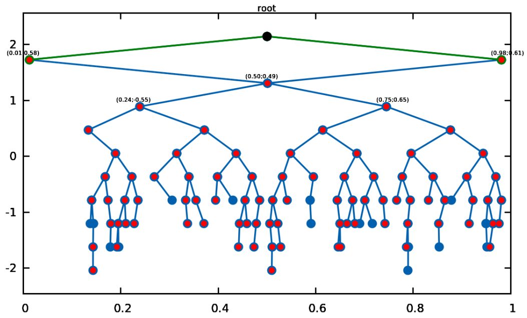

Figure 5: Constructed wavelet tree and extracted subtree.

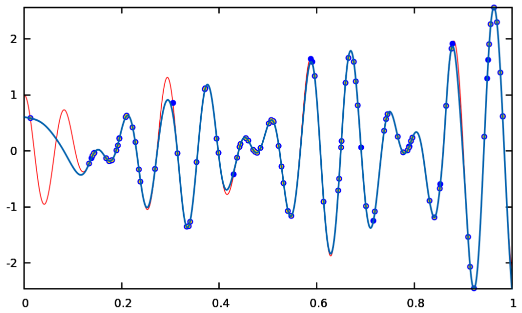

Figure 6: Interpolated function in red, result of the interpolation in blue.

6.4 Benching program

Benching code to check efficiency and check interpolation error in ℓ_2 and ℓ_∞ norm. See comments in the code.

Results of the run on my computer as well as some graphics are available in folder *results/*.

6.5 Animation program

Executable to generate incremental sample interpolation animations (generate gif files).

Overview : See *slides/img/interpolation.gif* for an example, see comments in the source code to generate your own gifs.

6.6 Source code overview

Here are the source files of the C++ implementation :

```
Root folder
 ├─ CMakeLists.txt ........................................ CMake configuration file for C++
 ├─ build/ ........................................................... Used to compile C++
 ├─ modules/ ............................................................... CMake helpers
 └─ src/ ................................................ Contains all C++ sources and headers
     ├─ main.cpp
     ├─ bench.cpp
     ├─ animation.cpp
     ├─ gnuplot ................................................ Contains gnuplot utilities
     │   ├─ gnuplot.hpp ................................ C++ header only gnuplot wrapper library
     │   ├─ affineTransformation.hpp
     │   ├─ plotBox.hpp
     │   └─ plotUtils.hpp
     ├─ sample ................................................ Contains sample structures
     │   ├─ point.hpp
     │   ├─ sample.hpp
     │   ├─ functionSample.hpp
     │   └─ randomSample.hpp
     ├─ tree ..................................................... Contains tree structures
     │   ├─ treeNode.hpp
     │   ├─ integerGrid.hpp
     │   ├─ binaryTreeNode.hpp
     │   └─ waveletTree.hpp
     ├─ wavelets ...................................... Contains wavelet related structures
     │   ├─ interval.hpp
     │   ├─ wavelet.hpp
     │   ├─ deslaurierDubuc.hpp
     │   ├─ deslaurierDubucUtils.hpp
     │   └─ waveletMapper.hpp
     └─ utils ................................ Mainly utilities : constants, vector, random, ...
         ├─ config.hpp.in ....................................... Cmake configured header input
         ├─ config.hpp.out ............ Cmake configured header output (created at compile time)
         ├─ consts.hpp
         ├─ defines.hpp
         ├─ globals.hpp
         ├─ headers.hpp
         ├─ utils.hpp
         ├─ maths ....................................................... Vector and Matrix
         │   ├─ vec.hpp
         │   ├─ vec2.hpp
         │   ├─ vec3.hpp
         │   ├─ vecBool.hpp
         │   └─ matrix.hpp
         └─ random ........................................... Random number generation utility
             └─ rand.hpp
```